Министерство образования Республики Беларусь

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра электронных вычислительных машин

ОТЧЕТ

О ЛАБОРАТОРНОЙ РАБОТЕ № 5

Шаблоны функций и классов

по дисциплине «Программирование на языках высокого уровня»

Выполнил ст. гр. 450503 А.П. Красько

Проверил асс. каф. ЭВМ И.Г. Скиба

Минск 2025

# 1 ПОСТАНОВКА ЗАДАЧИ

Создать параметризованный массив с конструкторами, деструктором и перегруженными операторами + , =.

# 2 ДИАГРАММА КЛАССОВ

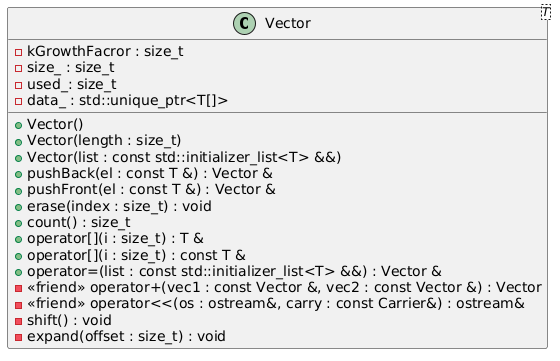


Рисунок 2.1 – Диаграмма классов

# 3 ЛИСТИНГ КОДА

Файл screens.hh:

#pragma once  
#include <memory>  
#include "vector.hh"  
namespace screens {  
void printMainScreen();  
bool inputVector(vec::Vector<double> &vec);  
bool addVectors(const vec::Vector<double> &vec1,const vec::Vector<double> &vec2);  
bool printVectors(const vec::Vector<double> &vec1,const vec::Vector<double> &vec2);  
} // namespace screens

Файл vector.hh:

#pragma once  
#include <exception>  
#include <format>  
#include <initializer\_list>  
#include <iostream>  
#include <memory>  
#include <ranges>  
namespace vec {  
template <class T>  
class Vector {  
 private:  
 constexpr static const size\_t kGrowthFacror = 2;  
 size\_t size\_;  
 size\_t used\_ = 0;  
 std::unique\_ptr<T[]> data\_;  
  
 private:  
 friend std::ostream &operator<<(std::ostream &os, const Vector &vec) {  
 os << '[';  
 for (size\_t i = 0; i < vec.used\_; i++) {  
 os << vec.data\_[i] << ' ';  
 }  
 os << "]";  
 return os;  
 }  
 friend Vector operator+(const Vector &vec1, const Vector &vec2) {  
 if (vec1.used\_ != vec2.used\_) throw std::invalid\_argument("Vector lengths mismatch");  
 Vector res(vec1.used\_);  
 for (size\_t i = 0; i < vec1.used\_; i++) {  
 res[i] = vec1.data\_[i] + vec2.data\_[i];  
 }  
 return res;  
 }  
  
 void shift() {  
 auto tmpList = std::make\_unique\_for\_overwrite<T[]>(size\_);  
 std::copy(data\_.get(), data\_.get() + used\_, tmpList.get() + 1);  
 data\_ = std::move(tmpList);  
 }  
 void expand(size\_t offset = 0) {  
 size\_ = size\_ \* kGrowthFacror;  
 if (size\_ == 0) size\_ = 1;  
 auto tmpList = std::make\_unique\_for\_overwrite<T[]>(size\_);  
 std::copy(data\_.get(), data\_.get() + used\_, tmpList.get() + offset);  
 data\_ = std::move(tmpList);  
 }  
  
 public:  
 Vector() : size\_{0}, data\_{nullptr} {};  
 explicit Vector(size\_t length) : size\_{length}, used\_{length}, data\_{std::make\_unique<T[]>(size\_)} {}  
 Vector(const std::initializer\_list<T> &&list)  
 : size\_{list.size()}, used\_{size\_}, data\_{std::make\_unique\_for\_overwrite<T[]>(size\_)} {  
 if (size\_ == 0) throw std::invalid\_argument("empty initializer\_list");  
 std::ranges::copy(list, data\_.get());  
 }  
  
 Vector &operator=(const std::initializer\_list<T> &&list) {  
 if (list.size() == 0) throw std::invalid\_argument("empty initializer\_list");  
 size\_ = list.size();  
 data\_ = std::move(std::make\_unique\_for\_overwrite<T[]>(size\_));  
 std::ranges::copy(list, data\_.get());  
 return \*this;  
 };  
  
 T &operator[](size\_t i) { return data\_[i]; }  
 const T &operator[](size\_t i) const { return data\_[i]; }  
  
 Vector &pushBack(const T &el) {  
 if (used\_ == size\_) expand();  
 data\_[used\_] = el;  
 used\_++;  
 return \*this;  
 }  
 Vector &pushFront(const T &el) {  
 if (used\_ == size\_)  
 expand(1);  
 else  
 shift();  
 data\_[0] = el;  
 used\_++;  
 return \*this;  
 }  
 void erase(size\_t index) {  
 if (index >= used\_) return;  
 T \*data = data\_.get();  
 std::copy(data + index + 1, data + used\_, data + index);  
 used\_--;  
 }  
 size\_t count() const { return used\_; }  
};  
  
} // namespace vec

Файл screens.cc:

#include <consoleUtils.hh>  
#include <l5/include/vector.hh>  
#include <limits>  
#include <memory>  
#include <print>  
#include <sstream>  
  
using namespace std;  
using namespace vec;  
using namespace console\_utils;  
  
namespace screens {  
void printMainScreen() {  
 auto [cols, rows] = getConsoleDimensions();  
 println("{:^{}}", "\x{1B}[48;5;35mLab 5\x{1B}[0m", cols);  
 println("Please select action:\n");  
 println(" 1.Input first vector");  
 println(" 2.Input second vector");  
 println(" 3.Print vectors");  
 println(" 4.Add");  
 println(" 5.Exit");  
}  
  
bool inputVector(Vector<double> &vec) {  
 string line;  
 print("Please enter vector (numbers separated by spaces): ");  
 getline(std::cin, line);  
 istringstream iss(line);  
 double num;  
 vec::Vector<double> tmpVec;  
 while (iss >> num) {  
 tmpVec.pushBack(num);  
 }  
 cin.clear();  
 cout << tmpVec;  
 vec = std::move(tmpVec);  
 return true;  
}  
bool addVectors(const Vector<double> &vec1, const Vector<double> &vec2) {  
 print("addition result: ");  
 try {  
 cout << vec1 + vec2 << endl;  
 } catch (const std::invalid\_argument &e) {  
 std::cerr << "Error: " << e.what() << '\n';  
 }  
  
 return true;  
}  
bool printVectors(const Vector<double> &vec1, const Vector<double> &vec2) {  
 cout << vec1 << endl;  
 cout << vec2 << endl;  
 return true;  
}  
} // namespace screens

Файл main.cc:

#include <consoleUtils.hh>  
#include <functional>  
#include <l5/include/vector.hh>  
#include <l5/include/screens.hh>  
  
  
#include <iostream>  
#include <memory>  
  
  
using namespace std;  
using namespace console\_utils;  
using namespace screens;  
  
int main(void) {  
 vec::Vector<double> vec1;  
 vec::Vector<double> vec2;  
 static array<function<bool()>, 5> actions = {  
 [&vec1]() { return inputVector(vec1); },  
 [&vec2]() { return inputVector(vec2); },  
 [&vec1, &vec2]() { return printVectors(vec1, vec2); },  
 [&vec1, &vec2]() { return addVectors(vec1, vec2); },  
 []() { return false; },  
  
 };  
 unsigned int response;  
 do {  
 printMainScreen();  
 readT(response, ">", [](unsigned int num) { return num > 0 && num <= 5; });  
 cout << "\x{1B}[2J\x{1B}[H\n";  
 } while (actions[response - 1]());  
  
 return 0;  
}

Файл consoleUtils.hh:

#pragma once  
#include <functional>  
#include <iostream>  
#include <limits>  
#include <iomanip>  
#include <chrono>  
namespace console\_utils {  
std::pair<int, int> getConsoleDimensions();  
  
  
template <typename T, typename CT>  
void readT(T& data, const std::string& massage, CT validator) {  
 std::cout << massage;  
 while (((std::cin >> data).fail()) || !validator(data)) {  
 std::cout << "Invalid input. Reread input requierments\n";  
 std::cin.clear();  
 std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
 std::cout << massage;  
 }  
 std::cin.clear();  
 std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
}  
template <typename T>  
void readT(T& data, const std::string& massage) {  
 std::cout << massage;  
 while ((std::cin >> data).fail()) {  
 std::cout << "Invalid input. Reread input requierments\n";  
 std::cin.clear();  
 std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
 std::cout << massage;  
 }  
 std::cin.clear();  
 std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
}  
template <typename T, typename CT>  
void readT(T& data, const std::string& message, CT validator, const std::string& errmess) {  
 std::cout << message;  
 while (((std::cin >> data).fail()) || !validator(data)) {  
 std::cout << errmess;  
 std::cin.clear();  
 std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
 std::cout << message;  
 }  
 std::cin.clear();  
 std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
}  
template <typename T = std::chrono::sys\_seconds, typename CT = const char \* >  
void readT(std::chrono::sys\_seconds &data, const char \* massage, const char \* format) {  
 std::cout << massage;  
 while ((std::cin >> std::chrono::parse(format, data)).fail()) {  
 std::cout << "Invalid input. Reread input requierments\n";  
 std::cin.clear();  
 std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
 std::cout << massage;  
 }  
 std::cin.clear();  
 std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
}  
} // namespace console\_utils

Файл consoleUtils.cc:

#include <iostream>  
  
#ifdef \_\_linux\_\_  
#include <sys/ioctl.h>  
#include <unistd.h>  
#endif  
  
#ifdef \_WIN32  
#include <Windows.h>  
#endif  
  
namespace console\_utils {  
std::pair<int, int> getConsoleDimensions() {  
#ifdef \_WIN32  
 CONSOLE\_SCREEN\_BUFFER\_INFO csbi;  
 GetConsoleScreenBufferInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &csbi);  
 return std::make\_pair(csbi.srWindow.Right - csbi.srWindow.Left + 1, csbi.srWindow.Bottom - csbi.srWindow.Top + 1);  
#endif  
#ifdef \_\_linux\_\_  
  
 struct winsize w;  
 ioctl(STDOUT\_FILENO, TIOCGWINSZ, &w);  
 return std::make\_pair(w.ws\_col, w.ws\_row);  
  
#endif  
}  
} // namespace console\_utils

**4 РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

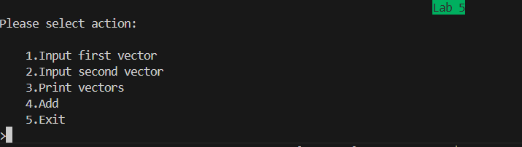


Рисунок 4.1 – Главное меню

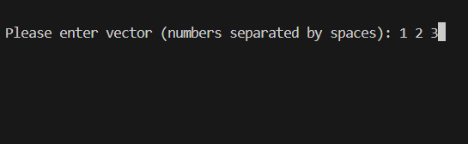


Рисунок 4.2 – Меню ввода массива

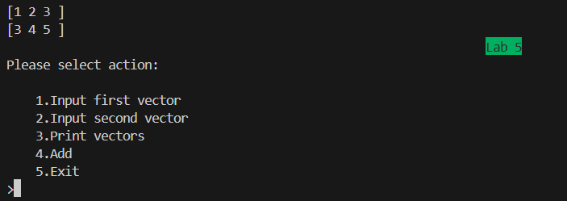


Рисунок 4.3 – Вывод массивов

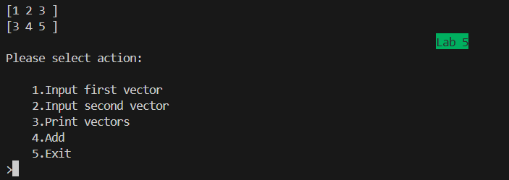


Рисунок 4.4 – Результат сложения

**5 ЗАКЛЮЧЕНИЕ**

В ходе выполнения лабораторной работы был разработан параметризованный класс Vector, реализующий динамический массив с поддержкой конструкторов, деструктора и перегруженных операторов + и =.